Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Jaime Elguero

# Design and development of a video-streaming recommendation system

Master's Thesis
Espoo, September 23rd, 2019

Supervisor: Professor Aristides Gionis
Advisor: Clemens Westrup

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

| | | | |
|---|---|---|---|
| **Author:** | Jaime Elguero | | |
| **Title:** | Design and development of a video-streaming recommendation system | | |
| **Date:** | September 23rd, 2019 | **Pages:** | v + 48 |
| **Major:** | Data Science | **Code:** | SCI3095 |
| **Supervisor:** | Professor Aristides Gionis | | |
| **Advisor:** | Clemens Westrup | | |

Recommendation systems are being widely adopted for two main reasons: address the limitations of search engines in the era of Big Data, and improve the user experience by helping the user find what they want but cannot express because it is hard to state as a query or because they do not even know it exists. In the case of video-on-demand services, this can be critical to keep users engaged and avoid churn.

The purpose of this project has been to develop and evaluate different recommendation algorithms to find what combination of factors can be used to simulate user behavior more accurately in a video-on-demand setting. Using the recommendation of the most popular items as a baseline, five different algorithms (with multiple parameter variations) have been tested. The results have shown that combining multiple simple models provides the best results, and that taking into consideration the context in which the recommendation is made can heavily improve the performance.

| | |
|---|---|
| **Keywords:** | recommendation system, video-on-demand, content-based filtering, collaborative filtering, hybrid filtering, matrix factorization, movies, information retrieval |
| **Language:** | English |

# Acknowledgements

This project is the result of several months of learning, experimenting and working that conclude my wonderful time at Aalto University.

Despite being the main contributor to this project, I would like to acknowledge a few people that with their guidance and support have made it possible to accomplish the objectives in due course.

I wish to thank Professor Aristides Gionis, for his guidance and feedback that have ensured a high quality level.

I wish to thank Clemens Westrup, for his advice to explore different solutions but also reminding me not to diverge to much too meet the schedule.

And I also want to thank Pasi Saari, Ville Renvall, and the whole data science team at Sanoma, for their continuous support.

Espoo, September 23rd, 2019

Jaime Elguero

# Contents

# Chapter 1

# Introduction

Search engines, once the main tool used to explore and find the right data within a database, are showing their limitations as the amount of data, from every domain, grows year after year [1]. Plain queries need to be complemented with additional information in order to efficiently retrieve the desired items. This additional information usually reflects the user's interests and the items' intrinsic characteristics, which are not only hard to state in a query, but also difficult to identify by the user.

Recommendation systems appear to extract these intrinsic relationships between users and items by analyzing the user's activity and the items' characteristics, and incorporate this new knowledge to the information retrieval process. The result is a better user experience, as the users take less time to decide what to watch, and discover new items they are likely to enjoy.

The purpose of this project is to develop a movie recommendation system for Ruutu [2], a Finnish video-on-demand service provided by Sanoma [3]. Sanoma is a media group based in Helsinki with more than 4,000 employes operating in 10 European countries. In Finland, Sanoma is the leading media company, with a weekly impact on 97% of the population through different channels: TV, radio, newspapers, etc.

## 1.1 Problem definition

The objective of this project is to evaluate different algorithms (and variations of several parameters) on how well they provide personalized recommendations for movies in a video-on-demand setting. This is done by evaluating the performance of these algorithms on an offline setup, and determining how well they compare to just recommending the most popular items.

### 1.1.1 Problem formalization

Let $U$ be the set of all $m$ users. Let $I$ be the set of all $n$ items. Let $g$ be a utility function that measures the usefulness of item $i$ to user $u$, e.g., $g : U \times I \rightarrow R$. Then for each user $u$ the goal is to choose such item $i' \in I$ that maximizes the user's utility:

$$\forall u \in U, i'_u = \underset{i \in I}{argmax}\ g(u, i) \tag{1.1}$$

This utility will be represented in the form of a *rating*. Therefore, $r_{ui}$ will be the rating given by user $u$ to item $i$, and the collection of all ratings given from each of the users to each of the items will be represented by $R \in \mathbb{R}^{m \times n}$.

The objective of this project is to find the utility function $g$ that better estimates these ratings, so that the recommended items are the ones the user finally chooses.

## 1.2 Thesis structure

The present document is organized in five chapters that cover specific parts of the project:

Chapter 1 offers an introduction to the thesis, explaining the rationale behind its existence and formally defining the problem to be solved.

Chapter 2 provides the reader with the fundamental principles behind recommendation systems, so that the reader has enough background to properly evaluate the results and the scope of the project.

Chapter 3 describes the methodology followed throughout the project, the data acquisition, analysis and preparation process, and the different algorithms tested and how they have been evaluated and compared.

Chapter 4 presents the results obtained for each of the experiments, making comparisons from an overall perspective to a more model-specific one.

Chapter 5 extracts the main lessons to be learned from the experiments, states what are the social and ethical implications that must be considered in a project like this, and suggests a few lines of future work.

Appendix A contains all the results obtained for each of the experiments in full extent.

# Chapter 2

# Background

The purpose of this chapter is to present a condensed but thorough overview of the theory of recommendation systems so that the reader may account with all the necessary information to properly understand and evaluate the scope and results of this project.

Therefore, the following sections will cover the most relevant aspects that must be considered, from the motivation and basics of this type of systems, to their limitations and challenges, as well as the state-of-the-art techniques widely adopted in the industry.

## 2.1 What is a recommendation system?

A recommendation (or recommender) system is a collection of techniques that filter the available information to find those items that will be most relevant to a specific user[4], thus, helping them make the most appropriate decision.

Traditionally, when facing a choice without sufficient experience or information, users have relied on recommendations made by others, usually in the form of word of mouth, reviews, or surveys[5]. However, when these sources of information are not available, users may find themselves lost and dissatisfied, especially as the number of options increases[6].

Recommendation systems arise to cover this information gap by analyzing the available information to reveal the latent relations between different users and different items. This way, the system may connect the experience of a specific user with others that have expressed similar interests (collaborative filtering), or it may connect different items by comparing their intrinsic characteristics (content-based filtering), or, maybe, a combination of both approaches (hybrid filtering). Likewise, information about the context in

which the choice is made, such as the location or time, can be of great value and improve the recommendations.

Recommendation systems have become very relevant in the recent years as search engines have proven themselves insufficient to find the most relevant items among an ever-growing collection of relevant items. This information overload requires a personalized approach that further filters the vast amount of data and content generated and stored every single day[1].

The task of the recommender system is to propose those items that are most likely to be enjoyed by the user. To do so, the system needs information about three different objects[7]: the *items* to be recommended, the *users* to whom the items are recommended, and the interactions between the users and the recommender system, also known as *transactions*.

The objective of the recommendation system is to estimate the ratings for all the possible user-item interactions to elaborate a ranking using these values, and make recommendations based on this ranking.

Figure 2.1 shows the overview of a recommender system. The system constructs a *profile* of each *user* based on individual traits such as the personal demographic information or the interaction with the system. Likewise, the system accounts with a *representation* of the *items* based on their intrinsic characteristics. These user and item representations are used by a *utility function* to determine what rating would the user give to each item. These values are used to elaborate a *ranking* of potential items that the user might enjoy, which are then *presented* to the user.

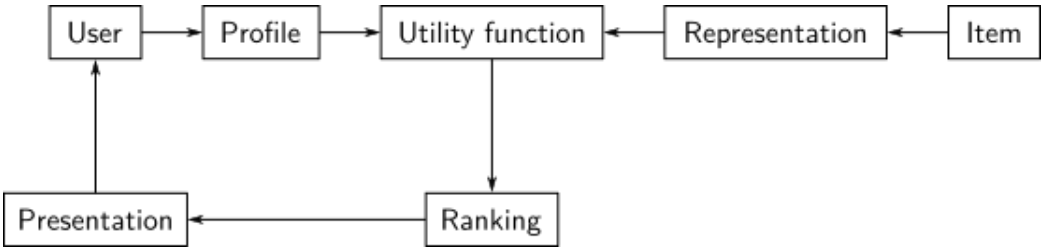


Figure 2.1: Recommendation system overview

## 2.2 Content-based filtering

Content-based filtering is one of the main techniques used by recommendation systems. It consists in establishing relational connections among different items using their intrinsic characteristics. This way the system is able to find and recommend similar items to the ones the user has already liked.

The first step in this process is to extract the items' features that are going to be compared. This can be done manually (e.g. manual labeling) or using an automatic algorithm.

Once the items' features have been extracted, different learning models are applied to identify and learn the underlying relationships that allow to identify similar items. These methods range from decision trees and neural networks to naive Bayes classifiers, being the cosine similarity one of the most relevant and widely adopted techniques.

Cosine similarity (explained in detail in section 3.3.3) consists in representing items in a vector space where each of the dimensions represents an item feature. This representation allows to determine the similarity between two items by comparing the direction of their respective vectors: the more parallel they are, the more similar.

There are, however, two important problems that this recommendation technique cannot address. First, the analysis of the content might be too limited, therefore failing to fully describe the item so it can be properly related to similar items. Second, as the system is designed to recommend similar items to those already liked by the user, it may "hide" different items that may also be of interest to the user. This is known as "overspecialization"[8].

## 2.3 Collaborative filtering

Collaborative filtering is a recommendation technique that consists in recommending items that have been liked by users with the same preferences as the target one.

Collaborative filtering can be classified into two main groups[8]:

- *Neighborhood (or memory-based, or heuristic-based) methods*, where the system finds the most relevant items using the ratings already stored in the system (from previous user-item interactions). The expected rating is calculated by considering the ratings given to that item by the user's neighbors (users with similar preferences).

- *Model-based methods*, where the system uses the recorded ratings to learn a predictive model, following a machine learning technique, being singular value decomposition (explained in detail in section 3.3.4) one of the most popular.

Collaborative filtering manages to overcome the overspecialization problem characteristic of content-based filtering, as it connects items through

users' preferences (domain independent) instead of items' characteristics (domain dependent).

However, collaborative filtering creates a problem that did not exist with content-based filtering methods. As the system only recommends items that have been liked by similar users, those new items with which the users have not interacted with will never be recommended. This is known as "cold start"[8].

## 2.4 Hybrid filtering

Hybrid filtering is a recommendation technique that seeks to overcome the individual problems of content-based filtering and collaborative filtering by using a combination of both techniques[9]. This way, the system will be able to provide accurate recommendations characteristic of collaborative filtering techniques while using content-based filtering to solve the cold start problem. Nowadays, most recommendation systems are built following this approach.

## 2.5 Video recommendations

One interesting and frequent application of recommendation systems is to help users find and discover video content that might be of their interest. Better recommendations lead to better user experience, which ultimately encourages users to consume more content. This is the reason why most video content providers such as video-on-demand services or video-centered social networks are implementing recommendation systems to their platforms.

Such is the case of Netflix[10], a video-on demand service that implements a recommendation system to personalize the user experience. It does so by providing enough diversity to address different moods, adapting to the user's preferences and explaining the recommendations[11].

YouTube[12] also implements a recommendation system to help users identify relevant videos within a massive repository where more than 24 hours of video are uploaded every minute[13].

## 2.6 User feedback

One important aspect of recommendation systems is that they are dynamic, that is, they learn from each interaction with the user and use this new knowledge to improve their recommendations. There are two ways in which the user can provide feedback to the system: explicitly and implicitly.

*Explicit feedback* is based on the users providing a rating to each item they have interacted with. This rating can be in the form of a discrete point scale (e.g. from 1 to 5) or a like/dislike option. Although this type of data better represents the users' interests, it shows two major drawbacks. First, it is hard to collect as it requires an effort from the user, who a lot times just wants to consume the content and does not bother about providing this feedback. And second, even if it the data was collected, the information would not be homogeneous as different users may have different criteria for rating items.

*Implicit feedback* does not require any effort from the user as it is based on analyzing the users' behavior and using the items they have interacted with as a signal of what they like. While it is a more "user-friendly" way of interaction, the system is not able to differentiate between what the user really likes, what he likes, or even what he dislikes.

## 2.7 Context

Just like recommendations vary depending on the user, they may also vary depending on the context. Context enriches the recommendations by providing additional dimensions to consider. For example, contextual information such as the time or location of a user can be used to recommend open restaurants nearby that fulfill the user's culinary preferences.

There are three ways to incorporate context to a recommender system[14]:

- *Contextual pre-filtering*, where the contextual information is used to select that data before it goes into the model.
- *Contextual post-filtering*, where the contextual information is used to adjust the recommendations made by the model.
- *Contextual modeling*, where the contextual information is incorporated into the model as another data dimension to consider.

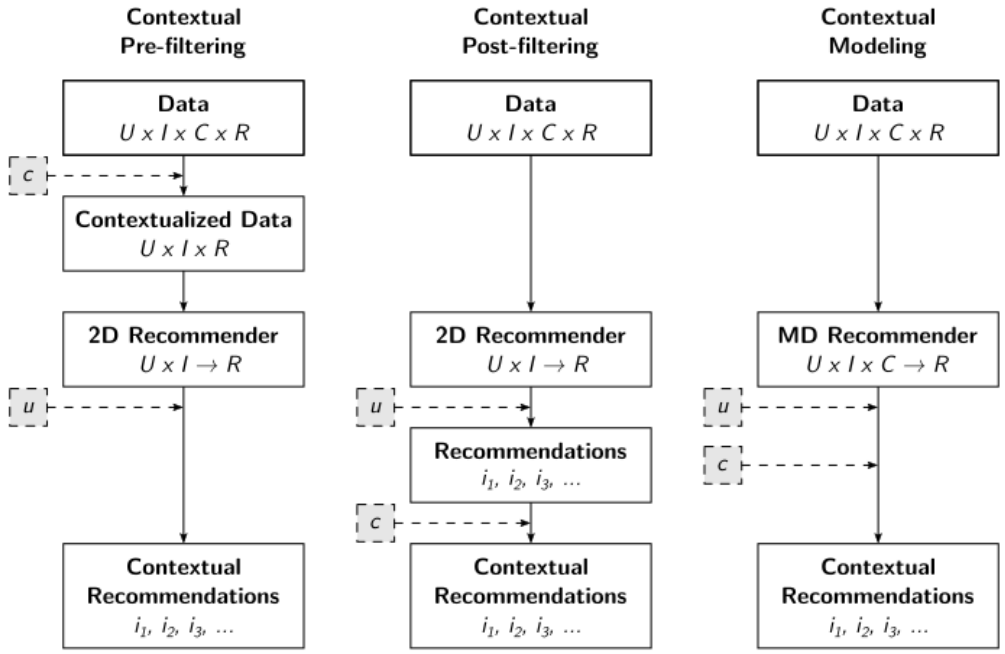


Figure 2.2: Paradigms for incorporating context in recommender systems[14].

Figure 2.2 shows at which point of the process are the specific user $u$ and context $c$ considered.

## 2.8 Evaluation

As many other computer applications, the recommendation problem can be solved implementing different algorithms and methods and, therefore, an evaluation system is required to identify those that better fulfill the requirements.

In the specific case of recommendation systems, there are three ways to evaluate the performance of the models[15]. Each type of evaluation has its advantages and disadvantages, and none of them is sufficient on its own:

- *Offline evaluation*, where the already collected data is used to simulate user behavior (e.g. evaluate how well the models predict the latest movie watched by the user). Offline evaluation is inexpensive as it does not require the collection of new data, but it is limited to measuring the prediction power of the models and can only partially simulate the behavior of the users, as the reasons behind a user choice are not fully available.

- *User studies*, where a set of subjects is analyzed and interviewed while interacting with the recommendation system. User studies provide a lot of information about the user experience and the quality of the recommendations, but they are expensive to conduct and are limited to a small sample of users.

- *Online evaluation*, where the recommendation system is implemented and analyzed, at least partially (e.g. not available to all the users), in a real environment. Online evaluation provides the most realistic results, but it involves a certain risk as a bad recommendation system may cost the confidence of the user. It is for this reason that offline evaluation is used to identify those models with the most promising performance and only test those online.

# Chapter 3

# Methodology

The purpose of this chapter is to present the methodology that has been followed to achieve the project's objectives effectively as well as efficiently.

First, the organization of the project is addressed, explaining the methodology followed for each level of abstraction. Second, the data is described and analyzed, and its preparation for the modeling phase is explained. Third, the algorithms used for this project are presented, explaining how they work and why they are relevant for this type of problem. Fourth, the chosen metrics used to evaluate and compare the different models are presented. And, finally, the list of hardware and software configurations used to run all the experiments is detailed.

## 3.1 Two levels of abstraction, two methodologies

The organization of this project presents two levels of abstraction: the overall process and the technical process, each of them requiring a different approach strategy. Instead of designing from scratch these strategies, two industry standards have been chosen, as they have proven to be both competent and powerful. Therefore, the overall process (the highest level of abstraction) has been designed using the Double Diamond design process, and the technical process has been designed using the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology.

### 3.1.1 Double Diamond

The Double Diamond model is a design process defined by the British Design Council[16] that proposes that the creative process should be divided into

four consecutive steps:

- *Discover*: focus on understanding the problem and the elements involved, gathering information and insights from all the perspectives.
- *Define*: analyze the information retrieved to find what aspects are the most important and decide where to focus to have a strong impact while remaining feasible.
- *Develop*: explore different solutions, creating prototypes and testing them.
- *Deliver*: using the solution that better fulfills the requirements, finalize and deliver the final product.

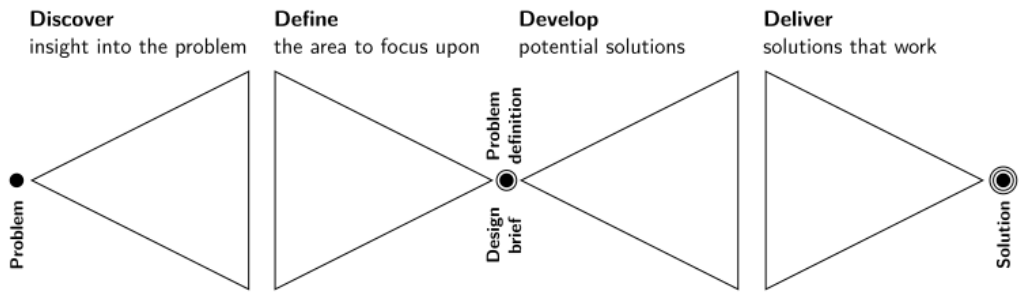


Figure 3.1: Double Diamond design process[16]

Such an organization of workflow allows to deeply understand the nature of the problem (Discover) without losing focus (Define), and explore the possible solutions (Develop) to find the one that better fulfills the project's objectives (Deliver). It is a great balance between creativity and efficiency.

In the case of this project, the Discovery phase consisted in a literature review of all the different techniques used in recommendation systems as well as the different aspects to consider. The Definition phase consisted in selecting those aspects and techniques that looked more promising to develop and test in the Develop phase. Finally, with the results at hand, the Deliver phase consisted in selecting the best model and implement it as a final product.

### 3.1.2 CRISP-DM

The Cross-Industry Standard for Data Mining (CRISP-DM)[17] is the most widely-used analytics process standard[18]. It addresses the data mining process by dividing it into six phases:

- *Business understanding*: this is the most important phase as it involves understanding the business perspective and motivations of the project. Once the business objectives are clear, and the project risks have been assessed, we can set the data mining objectives and develop a project plan.

- *Data understanding*: the purpose of this phase it to understand the data. It involves collecting the data, analyzing and exploring it, and verifying its quality and integrity.

- *Data preparation*: this phase focuses on preparing the data to be used in the modeling phase. It involves selecting the data to use, cleaning it, constructing new data, and integrating different data sources.

- *Modeling*: in this phase different modeling techniques are tested and optimized. It involves selecting the models to use, preparing a train and test dataset, building the models and assessing their performance.

- *Evaluation*: the purpose of this phase is to perform a thorough evaluation of the model and its construction to ensure the business objectives are met. Therefore, we evaluate the result and review the process to determine whether to process with the deployment or repeat the previous step to improve the performance or achieve new data mining goals.

- *Deployment*: this is the final phase of the process and it involves accomplishing all the required tasks to finalize the project, such as implementing a final product or producing a final report with all the learned insights.

One of the characteristics of CRISP-DM methodology is that it allows for an iterative learning process (figure 3.2), as it acknowledges that information is not completely available right from the start and that new learned insights may influence the decisions made in each of the steps.

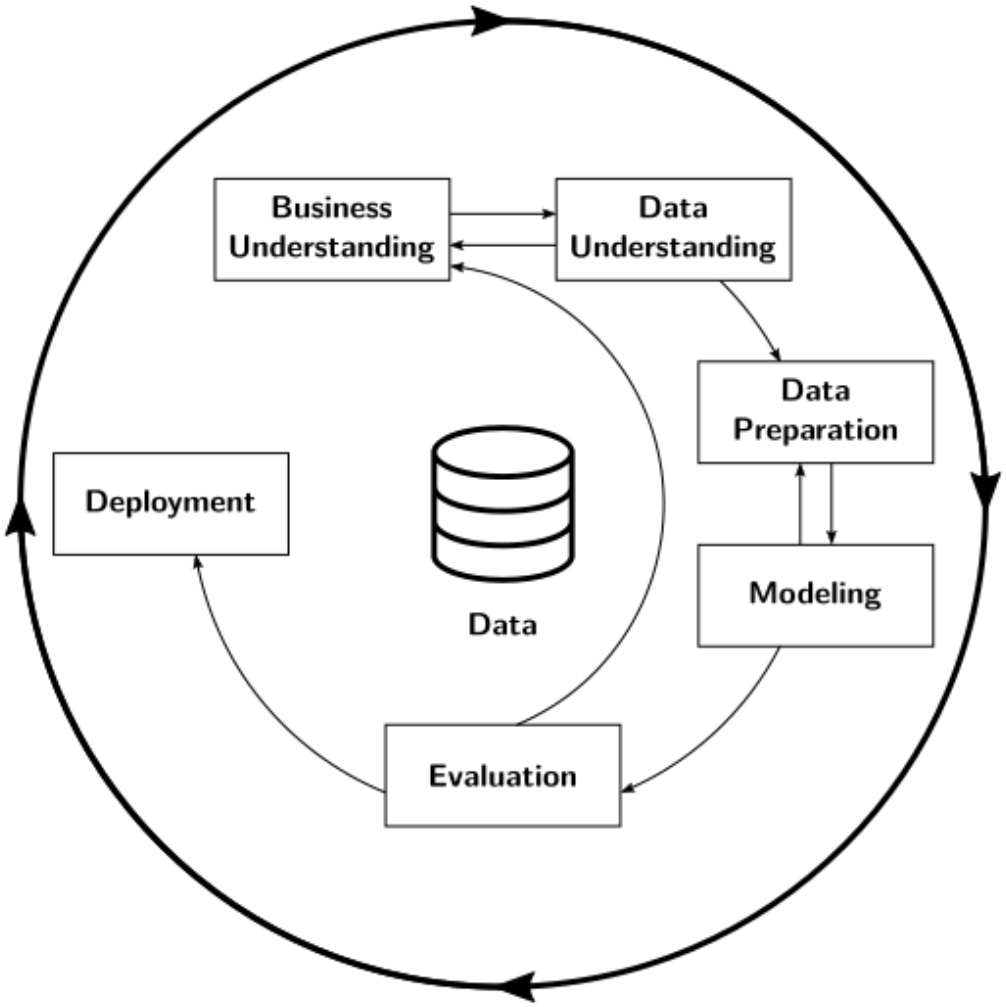


Figure 3.2: CRISP-DM process[17]

## 3.2 Data

This section presents and describes the data used for this project, from its sources and analysis, to its preparation for the modeling phase.

### 3.2.1 Data sources

The data used in this project has been retrieved from two different databases:

- The company's database, which contains information about the individual user sessions such as the movie watched, the duration of the

session or the type of user.

- The Internet Movie Database (IMDb)[19], which contains additional information about the movies, such as the actors or directors involved. This data can be easily retrieved by using the Open Movie Database (OMDb) API[20].

### 3.2.2 Data analysis

The data sample that has been retrieved from the company's database contains more than six million rows with information of user sessions between December 2017 and July 2019, involving more than one million users and over 400 movies. The data features include information about the session (ID, timestamp, duration), the user (ID, type) and the movies (ID, title, themes, length).

Likewise, the IMDb dataset contains information for each of the movies. Its features include information about the actors, the genre, the director, or the writer of the movie. Both datasets have been merged together to create a complete set of data.

The data does not include any user rating, so this is an implicit feedback problem where there is only information about whether a user-item interaction has occurred or not.

One representative characteristic of this type of data is the density, that is, the total number of unique user-item interactions with respect to the total number of possible interactions:

$$\text{density} = \frac{\text{user-item interactions}}{\text{users} \times \text{movies}} \tag{3.1}$$

The density of this specific dataset is 0.86%. This means that if the data was represented in a two dimensional matrix where each row would represent a user, each column would represent a movie, and the intersection of both would represent whether an interaction exists (1) or not (0), only 0.86% of the cells would have ones. The data is, therefore, very sparse.

### 3.2.3 Data preparation

In order to have the data ready for the modeling step, the data must first be preprocessed, that is, filtered and adjusted so that the algorithms can handle it effectively and efficiently.

As mentioned before, this is an implicit feedback dataset where there is only information about whether an interaction between a user and an item

has happened. However, there is also information about the duration of the session and the length of the movie, so these variables can be used to create a ratio that represents the percentage of the movie the user has watched. As a user may partially watch a movie in different sessions, this new variable needs to be aggregated in order to know the total percentage that has been watched. Once this is done, this variable can be binarized using a threshold to represent whether the user has watched the movie or not. For example, if a user has watched more than 50% of the movie, the value would be one, and zero otherwise.

Like many other video-on-demand platforms, the availability of movies varies through time: new movies become available while older movies leave the platform. This heterogeneity may create a problem where models recommend movies depending on their availability instead of their suitability for the users. To prevent this from happening only the movies that were available throughout a whole specific time interval have been filtered. In this case, focusing on the time interval between January and June 2018 led to more than a hundred available movies.

The platform currently allows four different types of users. The focus has been placed on those users that have a subscription, which is the dominant group with 49.02% of users and 74.83% of the user-item interactions. Also, those users that have watched less than five movies have been excluded, as under this threshold there is not enough information about the user.

The timestamp feature has been decomposed into year, month, day, day of the week, hour and minute, which will serve as contextual information in some of the models.

There are several variables that contain a set of labels as a value. In order to be able to work with such variables, they must first be one hot encoded, that is, a new variable must be created for each of these variables, and assigned a value of one or zero whether that label was present in the original variable or not, respectively. For example: {“genre”: “action, adventure”} becomes {“action”: 1, “adventure”: 1, “drama”: 0, “horror”: 0}.

As the normal functioning of the recommendation system is to predict what the user will watch next by analyzing what he has already watched, the data is split into the train and tests in a way that tries to replicate this behavior. Therefore, the test set contains the latest user-item interaction of each user, while the remaining interactions form the train set.

In order for the metrics to be reliable, the data has been randomly sampled in 100 groups of different users, and each of these groups has been used to train and test each of the models.

## 3.3 Algorithms

The purpose of this section is to present and describe the different algorithms that have been developed, implemented and evaluated.

### 3.3.1 Popularity (Baseline)

The popularity algorithm has been implemented to serve as a baseline with which to compare the performance of the other algorithms. It recommends the movies that have been watched by most users. Despite being completely unpersonalized, recommending popular movies is already a very good approach, as movie consumption appears to follow a Pareto distribution[21], that is, popular movies are very popular and will be watched by a lot of users, while unpopular movies are very unpopular and will be irrelevant for most users.

### 3.3.2 Popularity with context

As mentioned before, context may improve the quality of the recommendations as it provides additional dimensions to the data and, therefore, additional knowledge. One way to incorporate this contextual information is to adjust the popularity algorithm just described so that it recommends the most popular items under a specific context. For example, if a user is going to watch a movie on Christmas, recommend the movies that are being mostly watched during that period.

### 3.3.3 Cosine similarity

This content-based algorithm is based on representing the movies in a vector space where each dimension represents an attribute of the movie. Users, on the other hand, are represented by adding the vectors of the movies they have watched (figure 3.3).

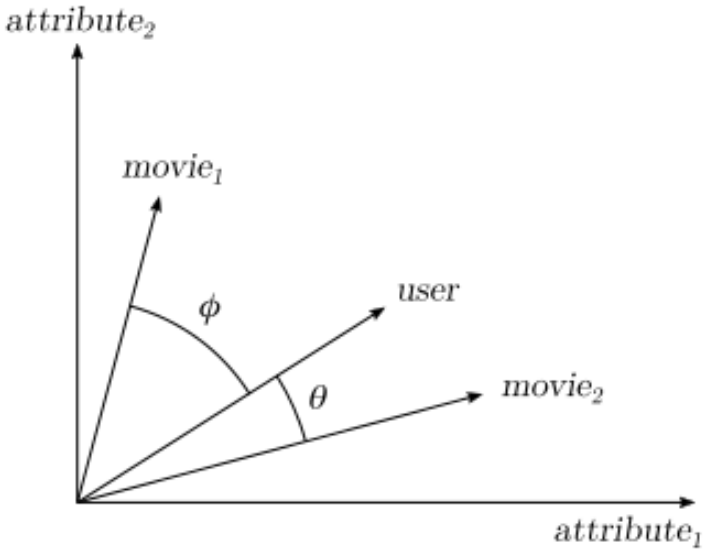


Figure 3.3: Vector space. Movie 2 is more similar to the user preferences since the angle between their respective vectors $\theta$ is smaller than the one between the user and movie 1 $\phi$.

The algorithm recommends movies that are similar to the user preferences, and it does so by finding movie vectors that are similar to the user vectors. To compute this similarity it uses the cosine similarity, which provides values between zero (perpendicular vectors, completely different) and one (parallel vectors, completely similar).

$$\text{cosine similarity}(\vec{u}, \vec{v}) = |cos(\theta)| = \left| \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|} \right| \tag{3.2}$$

For this project, the spatial dimensions represent the possible values of the movie themes, genres, actors, directors and writers. The model can incorporate these dimensions one a time or in different combinations.

Another movie attribute that has been used to find similar movies are the poster images. To do so, a convolutional neural network has been used to embed the images and represent them in a vector space where the cosine similarity can be computed. Instead of training a neural network from scratch, a VGG16[22] pretrained with the Imagenet[23] dataset has been implemented.

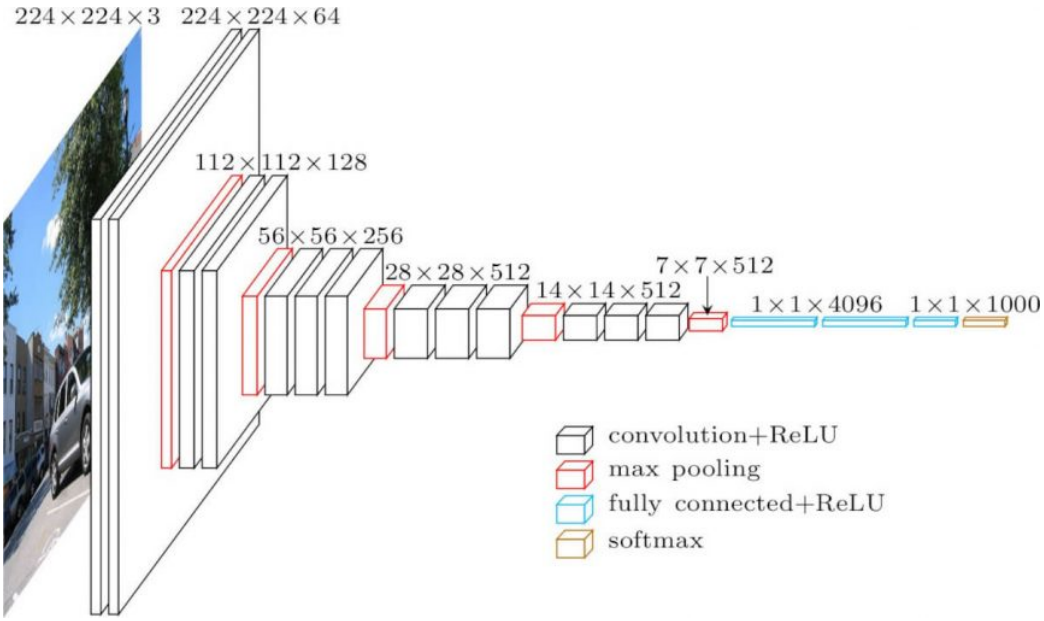


Figure 3.4: VGG16 structure[24]

### 3.3.4 Matrix factorization

Matrix factorization[25] is a latent factor model that by representing users and movies in a common latent space is able to identify the intrinsic relationships between different users and different movies.

It is based on the Singular Value Decomposition (SVD) method from linear algebra, where a matrix $A \in \mathbb{R}^{m \times n}$ is decomposed into the multiplication of three matrices $A = U\Sigma V^T$, with $U \in \mathbb{R}^{m \times f}$, $\Sigma \in \mathbb{R}^{f \times f}$, and $V \in \mathbb{R}^{n \times f}$, and where $f$ is the number of latent features.

This algorithm uses the SVD method to try to decompose the rating matrix $R \in \mathbb{R}^{m \times n}$ (where each row represents a user, each column represent a movie, and the intersection represents the rating given by the user to the movie) into the multiplication of two matrices $R = XY^T$, where $X \in \mathbb{R}^{m \times f}$ represents the latent features of each user, and $Y \in \mathbb{R}^{n \times f}$ represents the latent features of each movie. This way, the predicted ratings can be calculated as $\hat{r}_{ui} = x_u^T y_i$.

With this approach, a learning model can be created that determines $X$ and $Y$ by using the following cost function (with a regularizing term):

$$\min_{x_*, y_*} \sum_{r_{ui} \text{ is known}} (r_{ui} - x_u^T y_i)^2 + \lambda(||x_u||^2 + ||y_i||^2) \tag{3.3}$$

Once $X$ and $Y$ have been obtained, all the missing ratings from $R$ can

be computed, and used to recommend a set of movies to user $u$ by selecting the movies in decreasing order of $r_{ui}$.

This algorithm, originally designed for explicit feedback from the users, can be modified to work with implicit feedback[26]. In this case, instead of directly using the rating matrix, it is transformed into two new variables: preference and confidence.

The preference $p_{ui}$ indicates the preference of user $u$ to item $i$, and it is obtained by binarizing the $r_{ui}$ values:

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} \tag{3.4}$$

The confidence $c_{ui}$ represents the confidence in observing $p_{ui}$, and it grows as $r_{ui}$ grows:

$$c_{ui} = 1 + \alpha\, r_{ui} \tag{3.5}$$

where the constants $\alpha$ and $\epsilon$ control the rate of increase.

With the incorporation of these two new variables, the cost function becomes:

$$\min_{x_*, y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left( \sum_u ||x_u||^2 + \sum_i ||y_i||^2 \right) \tag{3.6}$$

### 3.3.5 Tensor factorization (iTALS)

One way to incorporate contextual information into a recommender system is by representing the data in a $D$-dimensional tensor $T$, where one dimension represents the users, another dimensions represents the movies, and the remaining $C$ dimensions represent the context under which each user-item interaction occurred.

Although there are different techniques to factorize, they do not usually scale very well. One of the algorithms that claims to have a nice computation performance is iTALS[27], *"a general ALS-based tensor factorization algorithm that scales linearly with the non-zero element of a dense tensor (when appropriate weighting is used) and cubically with the number of features"*.

As iTALS is designed to work with implicit feedback datasets, each element $T_{u,i,c_1,...,c_C}$ has a value of one if the user-item interaction has occurred under the specific context determined by $c_1, ..., c_C$, and a value of zero otherwise. Then, a weight matrix $W \in \mathbb{R}^D$ is created with the same dimensions as $T$, and where each element has a value of one if the corresponding element in

$T$ is zero, and a value greater than one otherwise. This matrix is equivalent to the confidence variable from matrix factorization.

The algorithm tries to decompose $T$ into $D$ matrices that when multiplied together using the Hadamard product reproduce the original tensor $T$:

$$\hat{T}_{i_1,i_2,...,i_D} = 1^T M^{(1)}_{\bullet,i_1} \circ M^{(2)}_{\bullet,i_2} \circ ... \circ M^{(D)}_{\bullet,i_D} \tag{3.7}$$

These low rank matrices $M^{(i)}$ of size $f \times S_i$ (where $S_i$ represents the size of $T$ in the $i^{\text{th}}$ dimension) represent the users, the movies, and each of the contextual variables in a latent space of $f$ dimensions, where intrinsic relationships can be found. The loss function in this case is:

$$L(M^{(1)}, ..., M^{(D)}) = \sum_{i_1=1,...,i_D=1}^{S_1,...,S_D} W_{i_1,...,i_D} \left( T_{i_1,...,i_D} - \hat{T}_{i_1,...,i_D} \right)^2 \tag{3.8}$$

### 3.3.6 Linear combination of models

One interesting alternative to elaborating complex models that try to consider all the different factors is to combine simpler models[11]. This way, each base model can focus on one specific area and share the learned insights with a "global" model. One straight forward implementation of such idea is to linearly combine the recommendations made by different models, that is, to create a "voting system" where each model assigns points to each movie, and the final recommendations are based on the sum of these points:

$$\text{final ranking} = \sum_{i}^{N} c_i \cdot \text{ranking from model}_i \tag{3.9}$$

where $c_i$ is the "voting power" of each model, and $N$ is the number of models.

## 3.4 Evaluation

As mentioned in section 2.8, despite its limitations offline evaluation provides a reasonable methodology to filter out underperforming models so only the most promising are evaluated in the online setting.

For this project, the following metrics have been computed for each of the models tested offline:

### 3.4.1 Recall@k

The ratio between the number of recommended movies that were relevant and the total number of movies the user actually watched. The value of $k$ indicates the number of movies recommended.

### 3.4.2 Precision@k

The ratio between the number of recommended movies that were relevant and the total number of recommended movies. The value of $k$ indicates the number of movies recommended.

### 3.4.3 Mean Average Precision (MAP)

The average precision value obtained for the top-k movies after each relevant movie has been retrieved.

### Example

A system recommends 5 movies to a user, out of which 2 (`M2` and `M5`) are actually watched. This is how the metrics are computed:

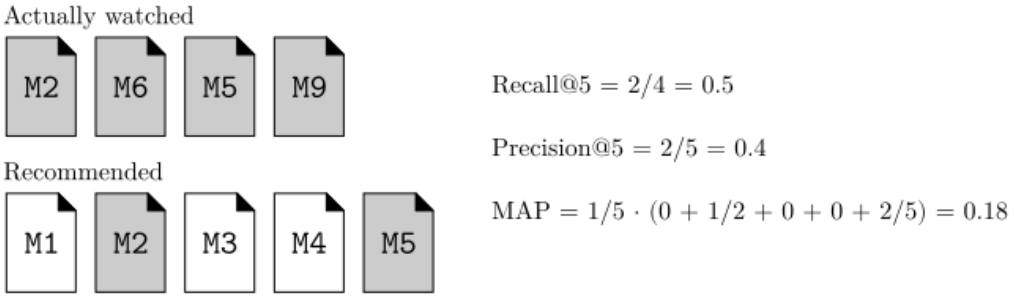


Figure 3.5: Example of metrics calculation.

## 3.5 Hardware and software

All the data mining process has been done with Python (version 3.7.3), complemented mainly with the following libraries: Psycopg (version 2.7.6.1) and Requests (version 2.22.0) for the data acquisition, Numpy (version 1.16.3) and Pandas (version 0.24.2) to manipulate the data, Keras (version 2.2.4)

and Tensorflow (version 1.14.0) to run the pretrained VGG16 convolutional neural network, and Matplotlib (version 3.1.0) to visualize the results.

The experiments of this project were run on a MacBook Pro with the following specifications:

- *Model*: MacBook Pro (13-inch, 2017, Two Thunderbolt 3 ports)
- *Processor*: 2,5 GHz Intel Core i7
- *Memory*: 16 GB 2133 MHz LPDDR3
- *Graphics*: Intel Iris Plus Graphics 640 1536 MB

# Chapter 4

# Results

The purpose of this chapter is to present and analyze the results obtained for each of the models and their variations.

As mentioned in section 3.2.3, the users have been assigned to 100 different groups, and the models and their variations have been trained and tested for each of these groups. The following results represent the aggregation of these tests with a 95% level of confidence.

Metrics Recall@k and Precision@k have been analyzed with $k = 20$, a typical value used in video recommendations as users do not usually see more than 20 elements of the recommended list [15].

It is important to mention that the meaning of the metrics might differ from the usual as the test set is composed of only one movie per user. This decision was made to be able to compare models that incorporated context as dimension with those that did not - context-based models would recommend different lists of movies for different contexts, while the rest of the models would recommend the same list. However, this deviation from the original meaning of the metric is not very important as the measured performance is compared to that of the baseline model.

The chapter is organized in a top-down style. First, a general overview of the performance of all the model variations is presented to identify which have the best results. Then, each model is analyzed to identify which variations have the best performance.

The complete list of numeric values is available in appendix A.

The names of the models and their variations have been encoded for visualization purposes. The corresponding code of each model can be found in Table 4.1.

| Code | Model |
|---|---|
| A | Popularity |
| B1 | Popularity with context (month) |
| B2 | Popularity with context (weekday) |
| B3 | Popularity with context (hour) |
| C1 | Cosine similarity (actors) |
| C2 | Cosine similarity (themes) |
| C3 | Cosine similarity (genre) |
| C4 | Cosine similarity (director) |
| C5 | Cosine similarity (writer) |
| C10 | Cosine similarity (actors, themes) |
| C11 | Cosine similarity (actors, genre) |
| C12 | Cosine similarity (actors, director) |
| C13 | Cosine similarity (actors, writer) |
| C14 | Cosine similarity (themes, genre) |
| C15 | Cosine similarity (themes, director) |
| C16 | Cosine similarity (themes, writer) |
| C17 | Cosine similarity (genre, director) |
| C18 | Cosine similarity (genre, writer) |
| C19 | Cosine similarity (director, writer) |
| C20 | Cosine similarity (actors, themes, genre) |
| C21 | Cosine similarity (actors, themes, director) |
| C22 | Cosine similarity (actors, themes, writer) |
| C23 | Cosine similarity (actors, genre, director) |
| C24 | Cosine similarity (actors, genre, writer) |
| C25 | Cosine similarity (themes, genre, director) |
| C26 | Cosine similarity (themes, genre, writer) |
| C27 | Cosine similarity (genre, director, writer) |
| C30 | Cosine similarity (actors, themes, genre, director) |
| C31 | Cosine similarity (actors, themes, genre, writer) |
| C32 | Cosine similarity (themes, genre, director, writer) |
| C40 | Cosine similarity (actors, themes, genre, director, writer) |
| D | Matrix factorization |
| E1 | Tensor factorization (month) |
| E2 | Tensor factorization (weekday) |
| E3 | Tensor factorization (hour) |
| M1 | Popularity with context (month) and cosine similarity (actors) |
| M2 | Popularity with context (month) and matrix factorization |
| M3 | Cosine similarity (all) and matrix factorization |
| M4 | Pop. with context (month) and cos. similarity (all) and matrix fact. |
| M5 | Matrix factorization and tensor factorization (month) |

Table 4.1: Model codes

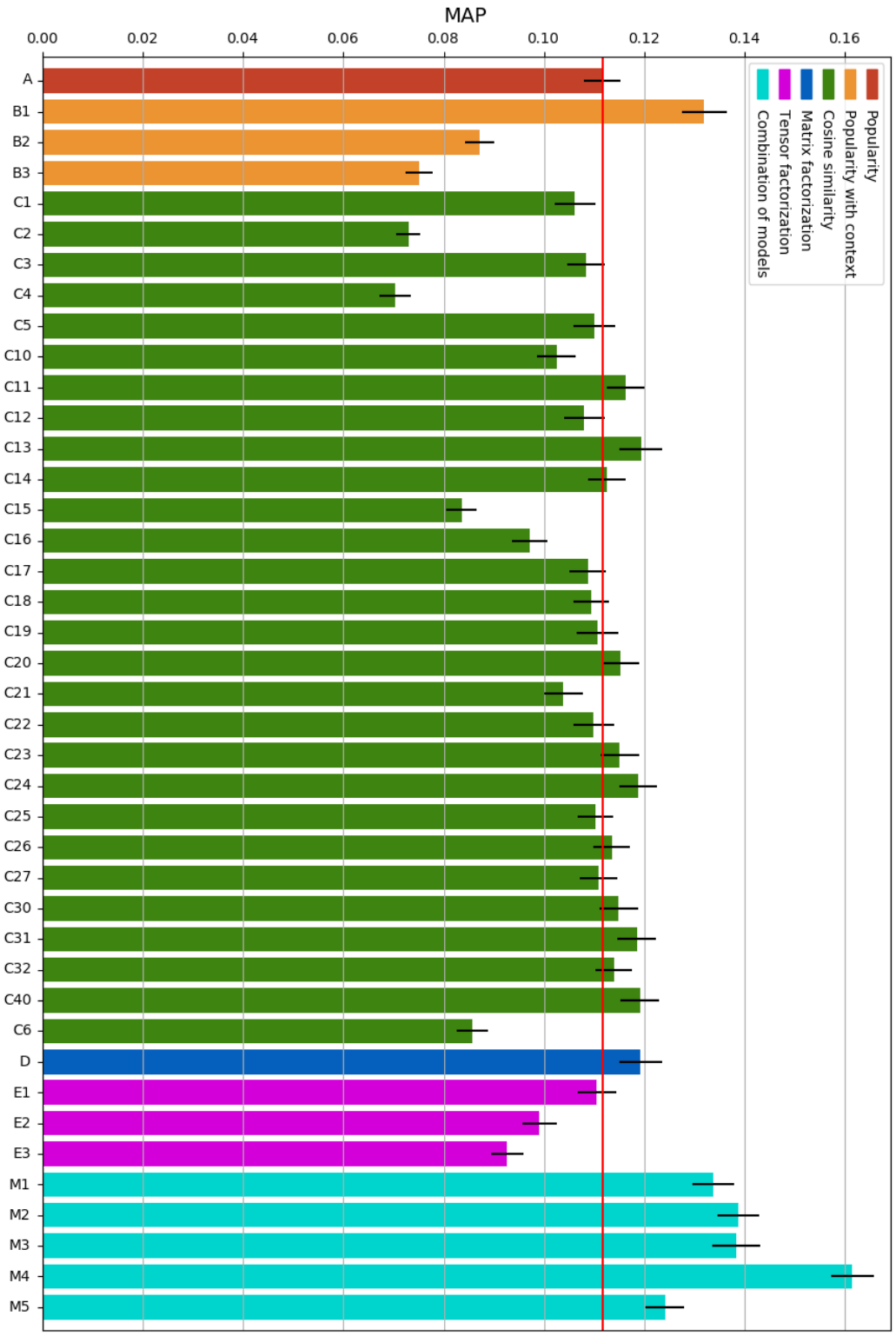


Figure 4.1: MAP. All models

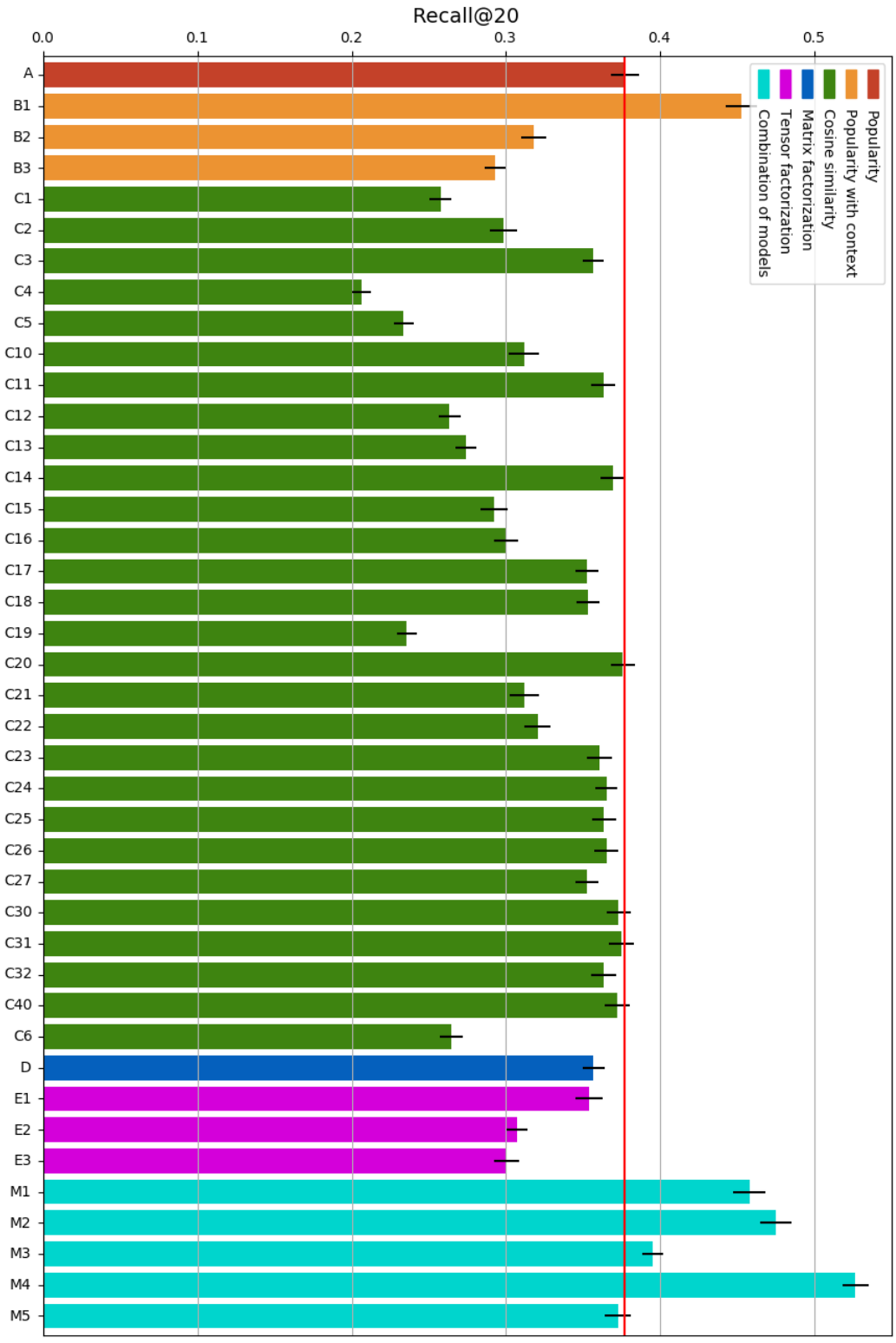


Figure 4.2: Recall@20. All models

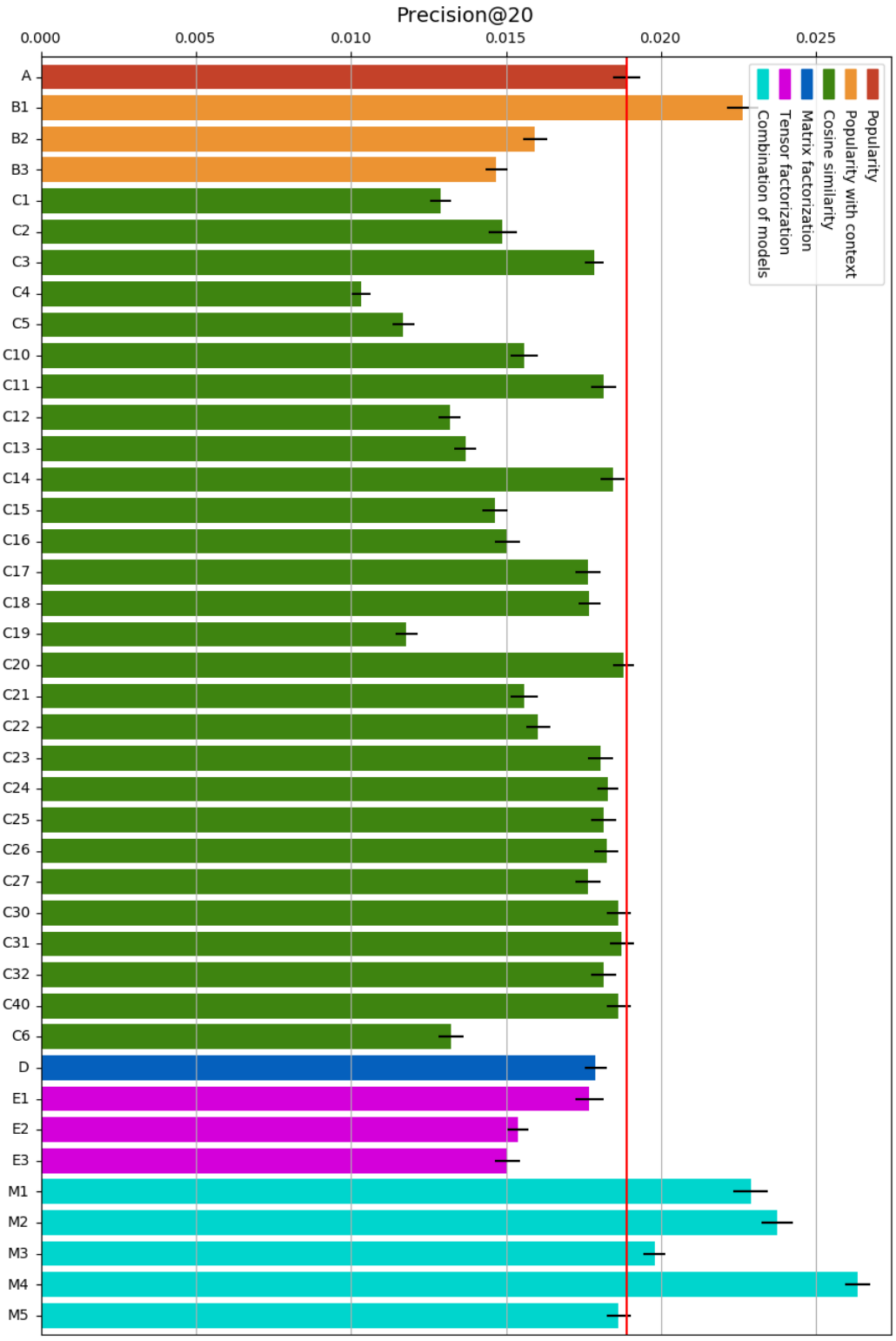


Figure 4.3: Precision@20. All models

## Overall

Figures 4.1, 4.2 and 4.3 present, respectively, the mean average precision (MAP), the Recall@20 and the Precision@20 values obtained for all the model variations: popularity (A), popularity with context (B1-B3), cosine similarity (C1-C40), matrix factorization (D), tensor factorization (E1-E3) and the combination of models (M1-M5). The vertical red line represents the value obtained for the popularity model, so that the remaining values can be easily compared to this baseline.

From the graphs it can be derived that it is not easy to outperform the popularity model. Only one variation of the popularity with context model (B1) and almost all of the combination of models (except M5) beat popularity in all metrics, being the combination of popularity with context (month), cosine similarity (all features) and the matrix factorization (M5) the variation with the best overall results.

## Popularity with context

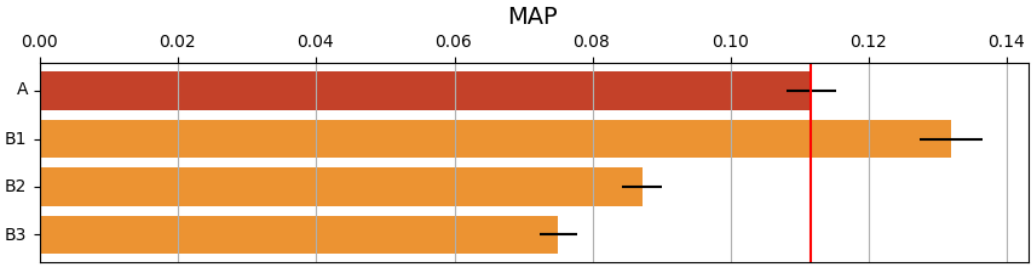


Figure 4.4: MAP. Popularity with context

Figure 4.4 presents the MAP values for each of the popularity with context model variations, that is, considering as context the month of the year (B1), the day of the week (B2), and the hour of the day (B3). As it can be derived from the graph, the month of year is a very powerful feature as it is the only context feature that manages to beat the popularity baseline.

## Cosine similarity

The cosine similarity variations that have been tested consist in using different movie attributes (actors, themes, genre, director and writer) and the combination of these features.

### One feature

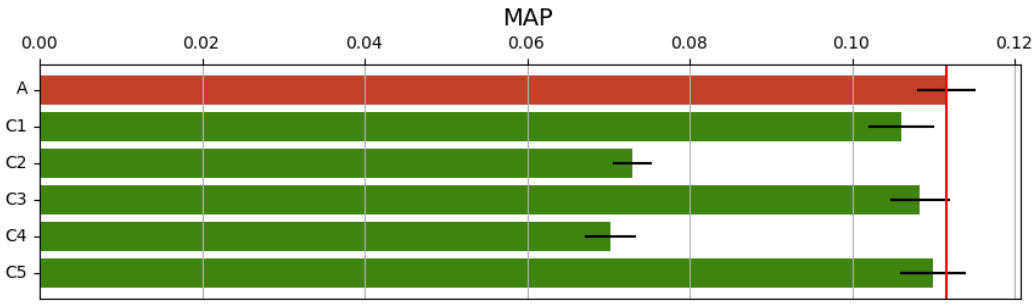


Figure 4.5: MAP. Cosine Similarity. One feature

As it can be seen in figure 4.5, none of the recommendations based on a single attribute can beat the popularity baseline, being the writer (C5), the genre (C3) and the actors (C1) the most relevant features.

### Two features

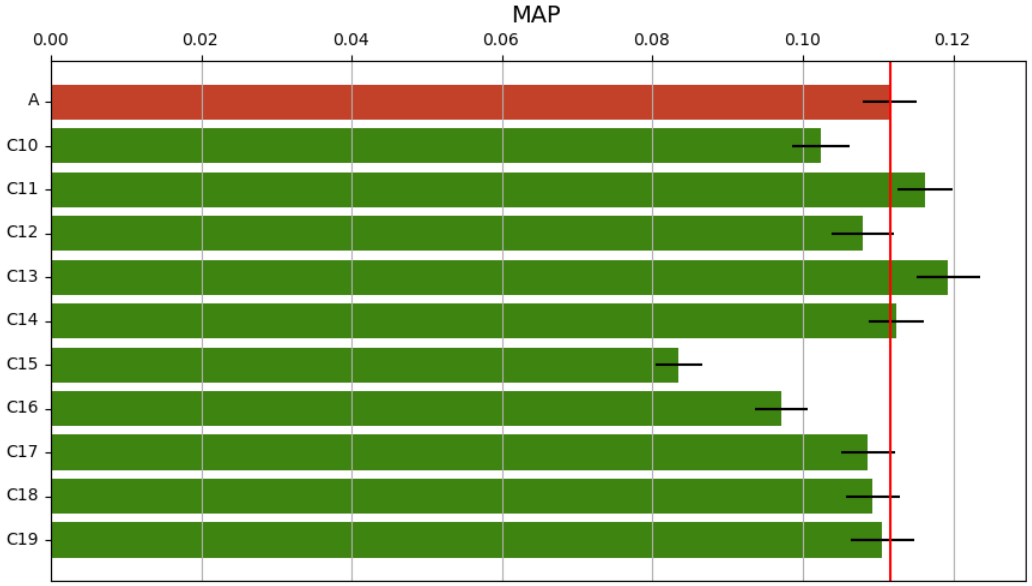


Figure 4.6: MAP. Cosine Similarity. Two features

By considering two attributes when making the recommendations, some of the model variations already beat the popularity baseline (Figure 4.6). Rec-

ommendations based on the movie's actors and writer (C13) has the best results, closely followed by those based on actors and genre (C11).

### Three or more features

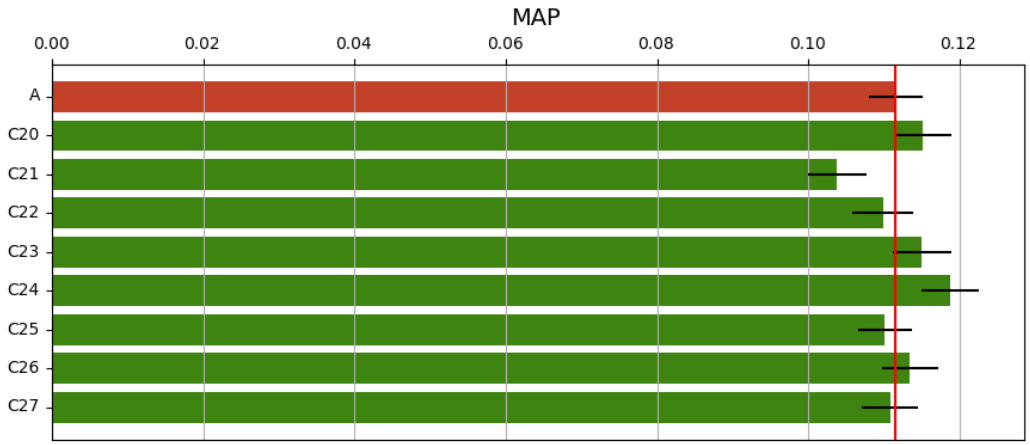


Figure 4.7: MAP. Cosine Similarity. Three features

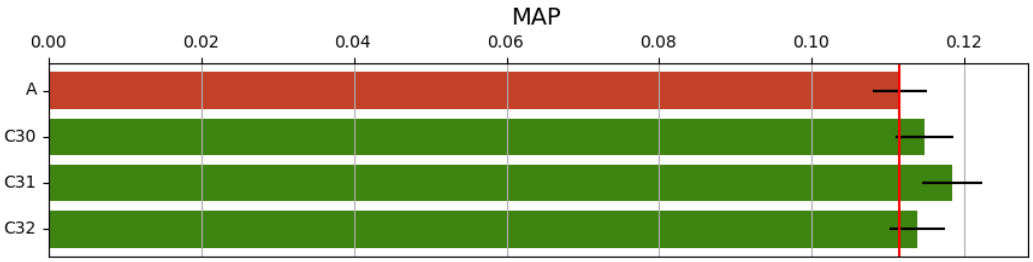


Figure 4.8: MAP. Cosine Similarity. Four features

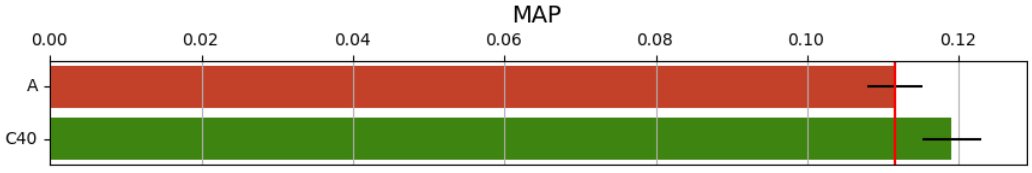


Figure 4.9: MAP. Cosine Similarity. All features

Figures 4.7, 4.8 and 4.9 show that including more features improves the overall performance (more model variations beat the popularity baseline),

but it does not manage to surpass the maximum obtained by two-feature model variation C13 (Figure 4.6). This means that recommendations based on the movie's actors and writer cannot be improved by adding the genre (C24), the genre and the themes (C31) or the genre, the themes and the director (C40), being these model variations the best of each group.

### Poster Images


Figure 4.10: MAP. Cosine Similarity. Poster images.

As shown in Figure 4.10, recommending movies that have similar poster images has proven to quite ineffective. Although the VGG16 network was able to identify similar images, the results show that users do not follow this criteria when selecting a movie.

## Matrix factorization

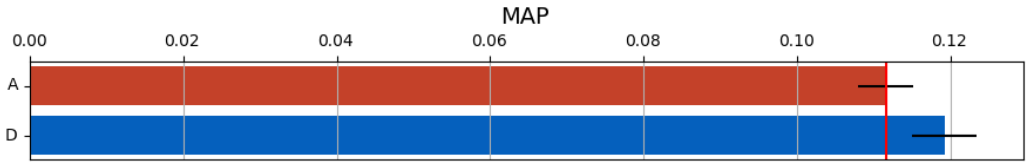


Figure 4.11: MAP. Matrix factorization.

Figure 4.11 shows that the model based on matrix factorization (D) surpasses the popularity baseline.

## Tensor factorization

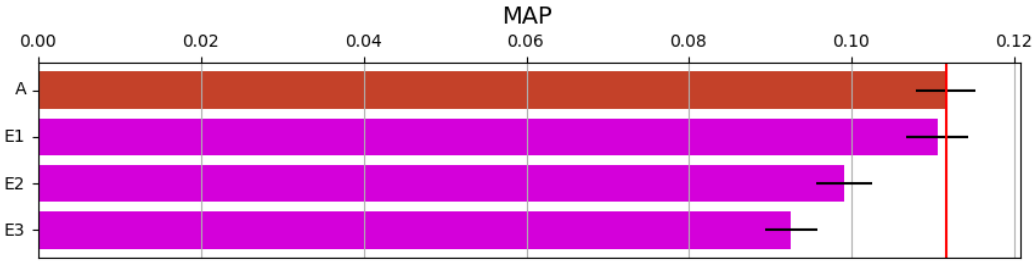


Figure 4.12: MAP. Tensor factorization.

As shown in Figure 4.12, the tensor factorization model variations do not manage to beat the popularity baseline. Like the popularity with context models (Figure 4.4), the month is the most relevant context feature.

## Combination of models

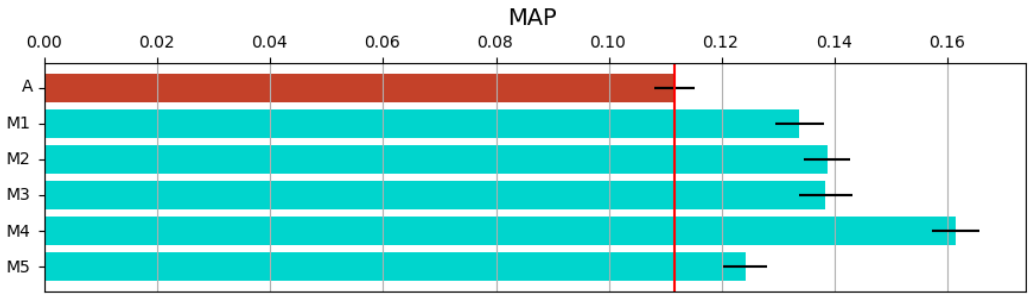


Figure 4.13: MAP. Combination of models.

Figure 4.13 shows that each and every combination of models that has been tested outperforms not only the popularity baseline but also the remaining model variations, being the linear combination of popularity with context (month), the cosine similarity considering all features and the matrix factorization, the model (M4) with the best performance.

Models presented in figure 4.13 use the coefficients that give each linear combination the best results. They have been identified by experimenting with different values. Figures 4.14, 4.15, 4.16, 4.17 and 4.18 show the different MAP values using different coefficients in each model variation.

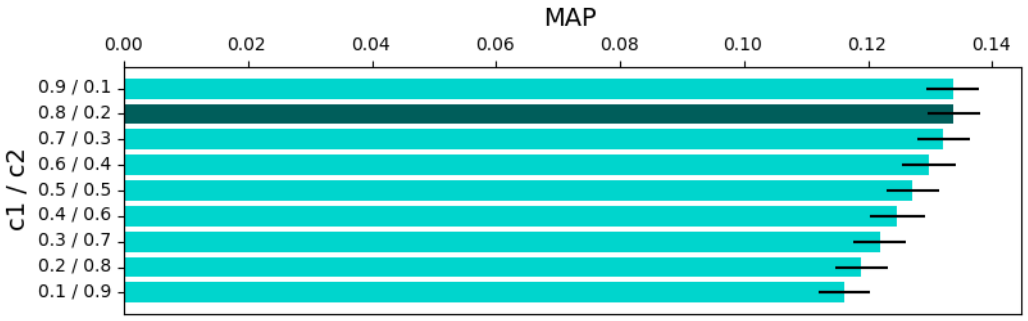


Figure 4.14: MAP. Combination of models with different coefficients (c1, c2): popularity with context (month) and cosine similarity (actors).

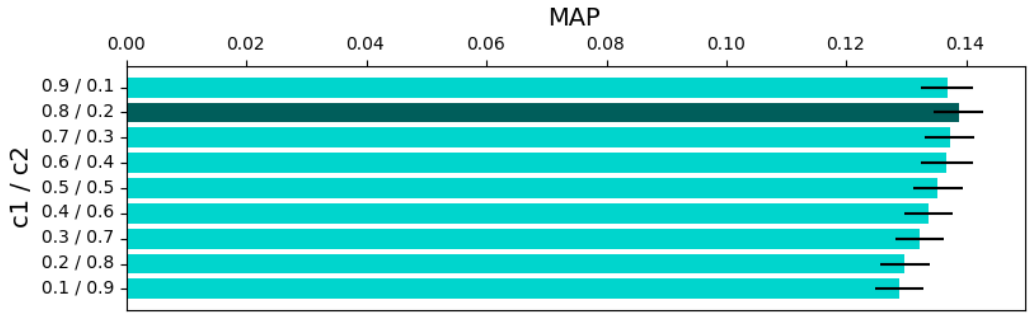


Figure 4.15: MAP. Combination of models with different coefficients (c1, c2): popularity with context (month) and matrix factorization.

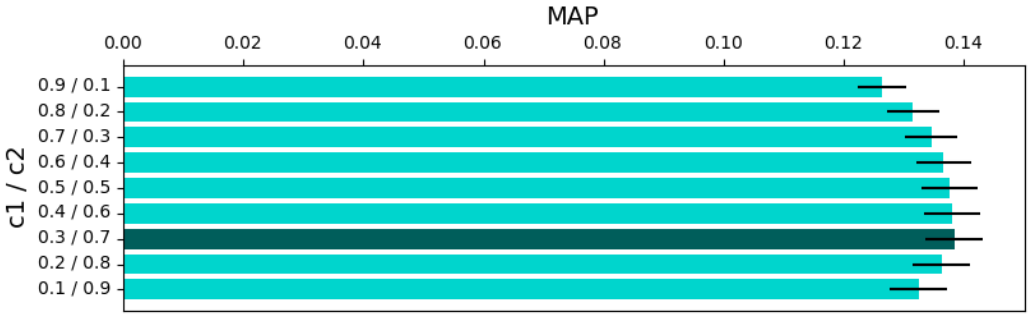


Figure 4.16: MAP. Combination of models with different coefficients (c1, c2): cosine similarity (all) and matrix factorization.

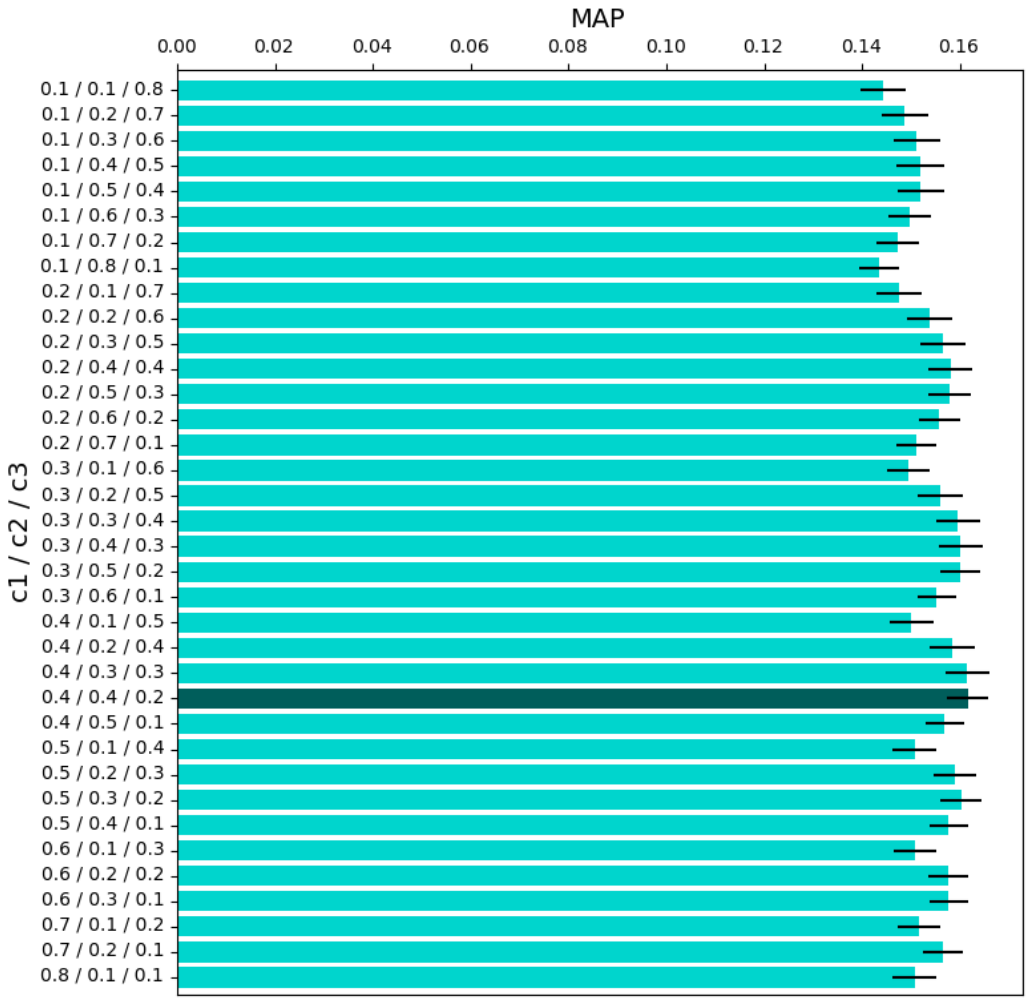


Figure 4.17: MAP. Combination of models with different coefficients (c1, c2, c3): popularity with context (month) and cosine similarity (all) and matrix factorization.

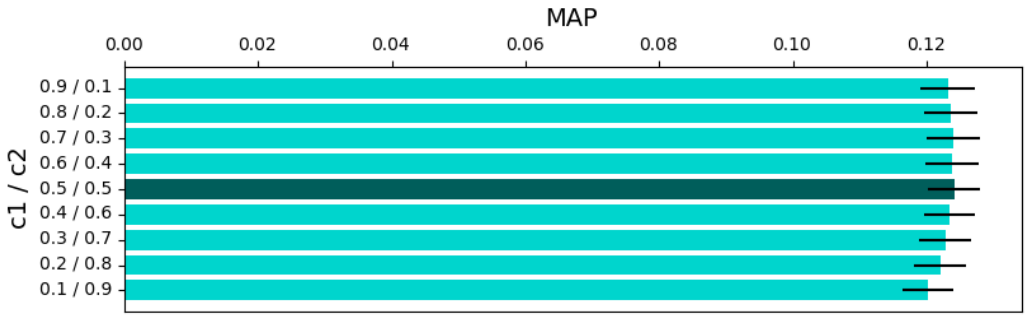


Figure 4.18: MAP. Combination of models with different coefficients (c1, c2): matrix factorization and tensor factorization (month).

# Chapter 5

# Discussion and conclusion

This final chapter brings this project to a close. It analyzes the results to extract the knowledge acquired from the experiments and summarizes it as lessons learned. It evaluates the impact of the recommendation system, considering its social and ethical implications. And, finally, it proposes a few lines of future work that might be interesting to develop.

## 5.1 Lessons learned

Here are presented the lessons learned from the experiments performed in this project:

- The popularity model is already hard to beat because it captures very well a frequent behavior of users: to watch movies that are popular and discard those that are not (Pareto distribution).
- If the popularity model is adjusted to consider the month (popularity with context model), the recommendations significantly improve. However, other contextual information such as the day of the week or the hour of the day do not perform as well.
- The cosine similarity model does not beat the popularity model when considering each attribute individually. However, the combination of these attributes does increase the performance, being the movie's actors and writer the attributes that contribute the most.
- Although VGG16 can easily identify similar poster images, the results show that user behavior does not follow such approach.

- Context has proven to bring a lot of value to the recommendations, as popularity with context (month) is the individual model with the best performance. However, when considering context as additional dimensions in a tensor (tensor factorization model), the performance decreases. This is probably due to a massive increase in sparsity with each dimension addition.

- Combining simple models specialized in extracting different aspects instead of a complex model that tries to capture everything has proven to be a great choice, as these models have had the best performance. The linear combination of the popularity with context (month), the cosine similarity (all) and the matrix factorization is the model with the best results.

## 5.2 Sustainability, ethics and societal impact

When implementing any engineering project, other impacts apart from the pure economical one must be considered. In this case, understanding the social implications is of great importance.

Although usually thought as a tool to help the users find what they want, recommendations can heavily determine the information the user has access to. In his book *The Filter Bubble*[28], Eli Pariser states that as recommendation systems become more and more personalized, they propose content that the user might like, filtering out a lot of content that the user will never know it even exists. This effect contributes to the question of how much filtering power should these algorithms have, which has become very relevant in social networks[29][30].

Better recommendations improve the user experience, which ultimately leads to a better engagement. However, leisure providers should have the moral responsibility of creating pleasing but not too addictive experiences.

Another important aspect that must be considered when implementing a recommendation system is privacy. As these systems need to use personal data to make the user profiles, developers must make sure that their solution is compliant with the General Data Protection Regulation (GDPR)[31], a European Union law implemented on May 2018 that seeks to protect the privacy of individuals.

The whole project has been developed using open source software. This approach has been followed for one main reason: this type of tools have an extensive support provided by the community of users, which facilitates the development and brings stability.

## 5.3 Future work

The discover phase of the project presented a lot of aspects of potential interest that could not be addressed due to time and resources limitations. These lines of future of work are summarized here.

- Personality is one of the key factors that influence users' preferences and could be a good solution for the cold-start problem. One way to incorporate user personality to a recommender system is by using the Five Factor Model (FFM) (also known by its acronym OCEAN), which evaluates the personality traits in terms of openness, conscientiousness, extraversion, agreeableness and neuroticism. By representing each user in a feature of space where each of these factors is a dimension, the system is able to make recommendations to each personality (e.g. people with high neuroticism are likely to watch movies from diverse directors). Despite being a field with a lot of potential, it is limited by the challenge of profile acquisition, as implicit acquisition is not reliable enough, and explicit questionnaires can be long and suspicious to the user[32].

- Another interesting factor to consider when making a recommendation is the user's mood. A nervous user might be more interested in an action movie, while a positive user might enjoy a romantic comedy[33].

- Content-based recommendations are usually based on content features such as the genre, the actors or the director. However, sometimes this list of factors can be limited or unavailable. Using the video's rich contents such as the audio or the motion can help overcome these limitations[34] (e.g. horror movies present similar audio patterns).

- One problem of implicit feedback data sets is that it cannot be strongly determined whether a user likes or dislikes an item - the systems usually interpret watching a movie as a positive signal. Finding new ways to infer explicit feedback from implicit feedback data would improve the confidence of these recommendations.

- A lot of research has been done on individual recommendations. However, group recommendations, although very interesting, have not been investigated as extensively. In the case of video-on-demand services, a group recommendation would consist in recommending the video content that better fulfills the interests of a group of people gathered to watch something together[35].

- One common challenge of recommendation systems in video-on-demand services is to identify the different people using the same user account, so that the recommendations can be adapted to each of them (e.g. children might consume a lot of cartoon content, which will unfortunately influence the recommendations made to their parents when its them who use the service).

# Bibliography


[1] Sarika Jain, Anjali Grover, Praveen Singh Thakur, and Sourabh Kumar Choudhary. Trends, problems and solutions of recommender system. *International Conference on Computing, Communication Automation*, page 955–958, 2015. doi: 10.1109/ccaa.2015.7148534.

[2] Ruutu. URL `https://www.ruutu.fi/`. (visited on 29.07.2019).

[3] Sanoma. URL `https://sanoma.com/`. (visited on 29.07.2019).

[4] Nayana Vaidya and A. R. Khachane. Recommender systems-the need of the ecommerce era. *IEEE 2017 International Conference on Computing Methodologies and Communication (ICCMC)*, page 100–104, 2017. doi: 10.1109/iccmc.2017.8282616.

[5] Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, Mar 1997. doi: 10.1145/245108.245121.

[6] Sheena S. Iyengar and Mark R. Lepper. When choice is demotivating: Can one desire too much of a good thing? *Journal of Personal and Social Psychology*, 79(6):995–1006, 2000. doi: 10.1037//0022-3514.79.6.995.

[7] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Introduction and challenges. In F. Ricci et al., editor, *Recommender Systems Handbook*, chapter 1, pages 1–34. Springer US, Boston, MA, 2015.

[8] Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In F. Ricci et al., editor, *Recommender Systems Handbook*, chapter 2, pages 37–76. Springer US, Boston, MA, 2015.

[9] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions.

*IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. doi: 10.1109/tkde.2005.99.

[10] Netflix. URL `https://www.netflix.com/`. (visited on 29.07.2019).

[11] Xavier Amatriain and Justin Basilico. Recommender systems in industry: A netflix case study. In F. Ricci et al., editor, *Recommender Systems Handbook*, chapter 11, pages 385–419. Springer US, Boston, MA, 2015.

[12] Youtube. URL `https://www.youtube.com/`. (visited on 29.07.2019).

[13] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *RecSys*, pages 293–296, 2010. doi: https://doi.org/10.1145/1864708.1864770.

[14] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In F. Ricci et al., editor, *Recommender Systems Handbook*, chapter 6, pages 191–226. Springer US, Boston, MA, 2015.

[15] Asela Gunawardana and Guy Shani. Evaluating recommender systems. In F. Ricci et al., editor, *Recommender Systems Handbook*, chapter 8, pages 265–308. Springer US, Boston, MA, 2015.

[16] The design process: What is the double diamond? URL `https://www.designcouncil.org.uk/news-opinion/design-process-what-double-diamond`. (visited on 30.06.2019).

[17] Colin Shearer. The crisp-dm model: The new blueprint for data mining. *Journal of Data Warehousing*, 5(4):13–22, 2000.

[18] Meta S. Brown. What it needs to know about the data mining process, Jul 2015. URL `https://www.forbes.com/sites/metabrown/2015/07/29/what-it-needs-to-know-about-the-data-mining-process/#3400258a515f`. (visited on 02.07.2019).

[19] Internet movie database (imdb). URL `https://www.imdb.com/`. (visited on 29.07.2019).

[20] Omdb api. URL `http://www.omdbapi.com/`. (visited on 29.07.2019).

[21] S. Sinha and S. Raghavendra. Hollywood blockbusters and long-tailed distributions. *The European Physical Journal B*, 42(2):293–296, 2004. doi: 10.1140/epjb/e2004-00382-7.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[23] Imagenet. URL `http://www.image-net.org/`. (visited on 29.07.2019).

[24] Neurohive. URL `https://neurohive.io/en/popular-networks/vgg16/`. (visited on 29.07.2019).

[25] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

[26] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. pages 263–272, 12 2008. doi: 10.1109/ICDM.2008.22.

[27] Balázs Hidasi and Domonkos Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. 09 2012. doi: 10.1007/978-3-642-33486-3_5.

[28] Eli Pariser. *The Filter Bubble*. Penguin Books, 80 Strand, London WC2R 0RL, England, 2011. ISBN 978-1-101-51512-9.

[29] Alfons López Tena. Twitter has gone from bastion of free speech to global censor, Jun 2017. URL https://www.businessinsider.com/twitter-has-gone-from-bastion-of-free-speech-to-global-censor-2017-6?IR=T (visited on 17.08.2019).

[30] L. Brent Bozell III. Facebook doesn't really believe in free speech. what they believe in (and actively practice) is censorship, Jul 2019. URL https://www.foxnews.com/opinion/facebook-doesnt-really-believe-in-free-speech-what-they-believe-in-and-actively-practice-is-censorship (visited on 17.08.2019).

[31] European Union. General data protection regulation (gdpr), May 2016. URL `http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG`. (visited on 02.09.2019).

[32] Marko Tkalcic and Li Chen. Personality and recommender systems. In F. Ricci et al., editor, *Recommender Systems Handbook*, chapter 21, pages 715–739. Springer US, Boston, MA, 2015.

[33] Pinata Winoto and Tiffany Tang. The role of user mood in movie recommendations. 37(8):6086–6092, 08 2010.

[34] Xingzhong Du, Hongzhi Yin, Ling Chen, Yang Wang, Yi Yang, and Xiaofang Zhou. Personalized video recommendation using rich contents from videos. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 12 2018.

[35] Judith Masthoff. Group recommender systems: Aggregation, satisfaction and group attributes. In F. Ricci et al., editor, *Recommender Systems Handbook*, chapter 22, pages 743–776. Springer US, Boston, MA, 2015.

# Appendix A

# Results

Tables A.1 to A.10 contain the test metrics (MAP, Recall@20 and Precision@20) obtained for each of the model variations, using 100 samples of randomly selected users without replacement. The values represent the confidence intervals with $\alpha = 0.05$.

| MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|
| low | up | low | up | low | up |
| 0.1078 | 0.1150 | 0.3672 | 0.3858 | 0.0184 | 0.0193 |

Table A.1: Results. Popularity.

| Context | MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|---|
| | low | up | low | up | low | up |
| month | 0.1272 | 0.1362 | 0.4420 | 0.4619 | 0.0221 | 0.0231 |
| weekday | 0.0841 | 0.0898 | 0.3091 | 0.3253 | 0.0155 | 0.0163 |
| hour | 0.0722 | 0.0775 | 0.2856 | 0.2993 | 0.0143 | 0.0150 |

Table A.2: Results. Popularity with context.

| Attributes | MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|---|
| | low | up | low | up | low | up |
| actors (A) | 0.1019 | 0.1100 | 0.2500 | 0.2639 | 0.0125 | 0.0132 |
| themes (T) | 0.0704 | 0.0752 | 0.2888 | 0.3062 | 0.0144 | 0.0153 |
| genre (G) | 0.1045 | 0.1119 | 0.3494 | 0.3626 | 0.0175 | 0.0181 |
| director (D) | 0.0669 | 0.0733 | 0.1993 | 0.2118 | 0.0100 | 0.0106 |
| writer (W) | 0.1057 | 0.1139 | 0.2265 | 0.2397 | 0.0113 | 0.0120 |
| A, T | 0.0984 | 0.1061 | 0.3013 | 0.3206 | 0.0151 | 0.0160 |
| A, G | 0.1124 | 0.1198 | 0.3548 | 0.3701 | 0.0177 | 0.0185 |
| A, D | 0.1037 | 0.1119 | 0.2559 | 0.2697 | 0.0128 | 0.0135 |
| A, W | 0.1149 | 0.1234 | 0.2665 | 0.2804 | 0.0133 | 0.0140 |
| T, G | 0.1086 | 0.1160 | 0.3608 | 0.3765 | 0.0180 | 0.0188 |
| T, D | 0.0803 | 0.0864 | 0.2830 | 0.3002 | 0.0142 | 0.0150 |
| T, W | 0.0935 | 0.1004 | 0.2915 | 0.3073 | 0.0146 | 0.0154 |
| G, D | 0.1049 | 0.1121 | 0.3441 | 0.3593 | 0.0172 | 0.0180 |
| G, W | 0.1056 | 0.1127 | 0.3451 | 0.3600 | 0.0173 | 0.0180 |
| D, W | 0.1062 | 0.1146 | 0.2284 | 0.2414 | 0.0114 | 0.0121 |
| A, T, G | 0.1112 | 0.1188 | 0.3672 | 0.3827 | 0.0184 | 0.0191 |
| A, T, D | 0.0998 | 0.1075 | 0.3017 | 0.3207 | 0.0151 | 0.0160 |
| A, T, W | 0.1057 | 0.1137 | 0.3115 | 0.3280 | 0.0156 | 0.0164 |
| A, G, D | 0.1111 | 0.1187 | 0.3519 | 0.3680 | 0.0176 | 0.0184 |
| A, G, W | 0.1149 | 0.1224 | 0.3571 | 0.3718 | 0.0179 | 0.0186 |
| T, G, D | 0.1064 | 0.1136 | 0.3550 | 0.3705 | 0.0177 | 0.0185 |
| T, G, W | 0.1096 | 0.1170 | 0.3564 | 0.3724 | 0.0178 | 0.0186 |
| G, D, W | 0.1070 | 0.1144 | 0.3443 | 0.3590 | 0.0172 | 0.0180 |
| A, T, G, D | 0.1109 | 0.1185 | 0.3647 | 0.3800 | 0.0182 | 0.0190 |
| A, T, G, W | 0.1144 | 0.1222 | 0.3658 | 0.3823 | 0.0183 | 0.0191 |
| T, G, D, W | 0.1101 | 0.1174 | 0.3547 | 0.3706 | 0.0177 | 0.0185 |
| A, T, G, D, W | 0.1150 | 0.1228 | 0.3636 | 0.3796 | 0.0182 | 0.0190 |
| Poster image | 0.0824 | 0.0886 | 0.2567 | 0.2712 | 0.0128 | 0.0136 |

Table A.3: Results. Cosine similarity.

| MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|
| low | up | low | up | low | up |
| 0.1148 | 0.1233 | 0.3491 | 0.3632 | 0.0175 | 0.0182 |

Table A.4: Results. Matrix factorization.

| Context | MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|---|
| | low | up | low | up | low | up |
| month | 0.1065 | 0.1142 | 0.3444 | 0.3623 | 0.0172 | 0.0181 |
| weekday | 0.0954 | 0.1024 | 0.2999 | 0.3133 | 0.0150 | 0.0157 |
| hour | 0.0892 | 0.0956 | 0.2918 | 0.3077 | 0.0146 | 0.0154 |

Table A.5: Results. Tensor factorization.

| Weights | | MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|---|---|
| $w_1$ | $w_2$ | low | up | low | up | low | up |
| 0.9 | 0.1 | 0.1292 | 0.1377 | 0.4490 | 0.4709 | 0.0225 | 0.0235 |
| 0.8 | 0.2 | 0.1293 | 0.1378 | 0.4468 | 0.4677 | 0.0223 | 0.0234 |
| 0.7 | 0.3 | 0.1277 | 0.1362 | 0.4331 | 0.4546 | 0.0217 | 0.0227 |
| 0.6 | 0.4 | 0.1252 | 0.1339 | 0.4006 | 0.4211 | 0.0200 | 0.0211 |
| 0.5 | 0.5 | 0.1227 | 0.1313 | 0.3656 | 0.3839 | 0.0183 | 0.0192 |
| 0.4 | 0.6 | 0.1201 | 0.1289 | 0.3373 | 0.3540 | 0.0169 | 0.0177 |
| 0.3 | 0.7 | 0.1174 | 0.1259 | 0.3242 | 0.3407 | 0.0162 | 0.0170 |
| 0.2 | 0.8 | 0.1144 | 0.1229 | 0.3070 | 0.3232 | 0.0153 | 0.0162 |
| 0.1 | 0.9 | 0.1119 | 0.1200 | 0.2778 | 0.2930 | 0.0139 | 0.0147 |

Table A.6: Results. Combination of popularity with context (month) and cosine similarity (actors).

| Weights | | MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|---|---|
| $w_1$ | $w_2$ | low | up | low | up | low | up |
| 0.9 | 0.1 | 0.1322 | 0.1410 | 0.4546 | 0.4747 | 0.0227 | 0.0237 |
| 0.8 | 0.2 | 0.1344 | 0.1426 | 0.4644 | 0.4842 | 0.0232 | 0.0242 |
| 0.7 | 0.3 | 0.1328 | 0.1412 | 0.4645 | 0.4842 | 0.0232 | 0.0242 |
| 0.6 | 0.4 | 0.1322 | 0.1408 | 0.4587 | 0.4761 | 0.0229 | 0.0238 |
| 0.5 | 0.5 | 0.1309 | 0.1392 | 0.4521 | 0.4692 | 0.0226 | 0.0235 |
| 0.4 | 0.6 | 0.1294 | 0.1375 | 0.4395 | 0.4576 | 0.0220 | 0.0229 |
| 0.3 | 0.7 | 0.1279 | 0.1361 | 0.4196 | 0.4366 | 0.0210 | 0.0218 |
| 0.2 | 0.8 | 0.1254 | 0.1336 | 0.3951 | 0.4111 | 0.0198 | 0.0206 |
| 0.1 | 0.9 | 0.1245 | 0.1326 | 0.3702 | 0.3853 | 0.0185 | 0.0193 |

Table A.7: Results. Combination of popularity with context (month) and matrix factorization.

| Weights | | MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|---|---|
| $w_1$ | $w_2$ | low | up | low | up | low | up |
| 0.9 | 0.1 | 0.1221 | 0.1302 | 0.3765 | 0.3915 | 0.0188 | 0.0196 |
| 0.8 | 0.2 | 0.1270 | 0.1356 | 0.3895 | 0.4034 | 0.0195 | 0.0202 |
| 0.7 | 0.3 | 0.1300 | 0.1387 | 0.3951 | 0.4095 | 0.0198 | 0.0205 |
| 0.6 | 0.4 | 0.1318 | 0.1409 | 0.3983 | 0.4134 | 0.0199 | 0.0207 |
| 0.5 | 0.5 | 0.1328 | 0.1421 | 0.3967 | 0.4121 | 0.0198 | 0.0206 |
| 0.4 | 0.6 | 0.1332 | 0.1424 | 0.3956 | 0.4107 | 0.0198 | 0.0205 |
| 0.3 | 0.7 | 0.1334 | 0.1429 | 0.3879 | 0.4013 | 0.0194 | 0.0201 |
| 0.2 | 0.8 | 0.1313 | 0.1407 | 0.3750 | 0.3890 | 0.0187 | 0.0195 |
| 0.1 | 0.9 | 0.1275 | 0.1370 | 0.3625 | 0.3765 | 0.0181 | 0.0188 |

Table A.8: Results. Combination of cosine similarity (all) and matrix factorization.

| Weights | | MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|---|---|
| $w_1$ | $w_2$ | low | up | low | up | low | up |
| 0.9 | 0.1 | 0.1189 | 0.1271 | 0.3521 | 0.3692 | 0.0176 | 0.0185 |
| 0.8 | 0.2 | 0.1194 | 0.1275 | 0.3598 | 0.3766 | 0.0180 | 0.0188 |
| 0.7 | 0.3 | 0.1198 | 0.1278 | 0.3551 | 0.3723 | 0.0178 | 0.0186 |
| 0.6 | 0.4 | 0.1196 | 0.1277 | 0.3583 | 0.3747 | 0.0179 | 0.0187 |
| 0.5 | 0.5 | 0.1201 | 0.1278 | 0.3634 | 0.3803 | 0.0182 | 0.0190 |
| 0.4 | 0.6 | 0.1195 | 0.1271 | 0.3655 | 0.3821 | 0.0183 | 0.0191 |
| 0.3 | 0.7 | 0.1187 | 0.1265 | 0.3647 | 0.3821 | 0.0182 | 0.0191 |
| 0.2 | 0.8 | 0.1180 | 0.1258 | 0.3637 | 0.3813 | 0.0182 | 0.0191 |
| 0.1 | 0.9 | 0.1162 | 0.1239 | 0.3554 | 0.3732 | 0.0178 | 0.0187 |

Table A.9: Results. Combination of matrix factorization and tensor factorization (month).

| Weights | | | MAP | | Recall@20 | | Precision@20 | |
|---|---|---|---|---|---|---|---|---|
| $w_1$ | $w_2$ | $w_3$ | low | up | low | up | low | up |
| 0.1 | 0.1 | 0.8 | 0.1394 | 0.1486 | 0.3901 | 0.4051 | 0.0195 | 0.0203 |
| 0.1 | 0.2 | 0.7 | 0.1437 | 0.1532 | 0.4075 | 0.4216 | 0.0204 | 0.0211 |
| 0.1 | 0.3 | 0.6 | 0.1461 | 0.1557 | 0.4199 | 0.4355 | 0.0210 | 0.0218 |
| 0.1 | 0.4 | 0.5 | 0.1467 | 0.1565 | 0.4264 | 0.4410 | 0.0213 | 0.0221 |
| 0.1 | 0.5 | 0.4 | 0.1470 | 0.1565 | 0.4295 | 0.4444 | 0.0215 | 0.0222 |
| 0.1 | 0.6 | 0.3 | 0.1450 | 0.1539 | 0.4335 | 0.4474 | 0.0217 | 0.0224 |
| 0.1 | 0.7 | 0.2 | 0.1426 | 0.1514 | 0.4321 | 0.4467 | 0.0216 | 0.0223 |
| 0.1 | 0.8 | 0.1 | 0.1390 | 0.1472 | 0.4161 | 0.4314 | 0.0208 | 0.0216 |
| 0.2 | 0.1 | 0.7 | 0.1427 | 0.1518 | 0.4173 | 0.4327 | 0.0209 | 0.0216 |
| 0.2 | 0.2 | 0.6 | 0.1488 | 0.1581 | 0.4413 | 0.4560 | 0.0221 | 0.0228 |
| 0.2 | 0.3 | 0.5 | 0.1517 | 0.1608 | 0.4528 | 0.4671 | 0.0226 | 0.0234 |
| 0.2 | 0.4 | 0.4 | 0.1533 | 0.1623 | 0.4610 | 0.4755 | 0.0230 | 0.0238 |
| 0.2 | 0.5 | 0.3 | 0.1533 | 0.1620 | 0.4673 | 0.4813 | 0.0234 | 0.0241 |
| 0.2 | 0.6 | 0.2 | 0.1513 | 0.1597 | 0.4685 | 0.4828 | 0.0234 | 0.0241 |
| 0.2 | 0.7 | 0.1 | 0.1467 | 0.1548 | 0.4556 | 0.4704 | 0.0228 | 0.0235 |
| 0.3 | 0.1 | 0.6 | 0.1447 | 0.1536 | 0.4485 | 0.4643 | 0.0224 | 0.0232 |
| 0.3 | 0.2 | 0.5 | 0.1512 | 0.1604 | 0.4670 | 0.4830 | 0.0234 | 0.0242 |
| 0.3 | 0.3 | 0.4 | 0.1548 | 0.1638 | 0.4833 | 0.4971 | 0.0242 | 0.0249 |
| 0.3 | 0.4 | 0.3 | 0.1554 | 0.1643 | 0.4978 | 0.5127 | 0.0249 | 0.0256 |
| 0.3 | 0.5 | 0.2 | 0.1556 | 0.1638 | 0.4984 | 0.5143 | 0.0249 | 0.0257 |
| 0.3 | 0.6 | 0.1 | 0.1510 | 0.1589 | 0.4892 | 0.5042 | 0.0245 | 0.0252 |
| 0.4 | 0.1 | 0.5 | 0.1453 | 0.1544 | 0.4642 | 0.4807 | 0.0232 | 0.0240 |
| 0.4 | 0.2 | 0.4 | 0.1534 | 0.1628 | 0.4904 | 0.5059 | 0.0245 | 0.0253 |
| 0.4 | 0.3 | 0.3 | 0.1568 | 0.1657 | 0.5098 | 0.5255 | 0.0255 | 0.0263 |
| 0.4 | 0.4 | 0.2 | 0.1571 | 0.1655 | 0.5177 | 0.5343 | 0.0259 | 0.0267 |
| 0.4 | 0.5 | 0.1 | 0.1527 | 0.1605 | 0.5057 | 0.5221 | 0.0253 | 0.0261 |
| 0.5 | 0.1 | 0.4 | 0.1458 | 0.1550 | 0.4790 | 0.4959 | 0.0239 | 0.0248 |
| 0.5 | 0.2 | 0.3 | 0.1542 | 0.1630 | 0.5092 | 0.5266 | 0.0255 | 0.0263 |
| 0.5 | 0.3 | 0.2 | 0.1558 | 0.1641 | 0.5227 | 0.5397 | 0.0261 | 0.0270 |
| 0.5 | 0.4 | 0.1 | 0.1536 | 0.1613 | 0.5151 | 0.5318 | 0.0258 | 0.0266 |
| 0.6 | 0.1 | 0.3 | 0.1462 | 0.1549 | 0.4892 | 0.5088 | 0.0245 | 0.0254 |
| 0.6 | 0.2 | 0.2 | 0.1532 | 0.1615 | 0.5103 | 0.5290 | 0.0255 | 0.0264 |
| 0.6 | 0.3 | 0.1 | 0.1534 | 0.1614 | 0.5133 | 0.5321 | 0.0257 | 0.0266 |
| 0.7 | 0.1 | 0.2 | 0.1471 | 0.1556 | 0.4908 | 0.5108 | 0.0245 | 0.0255 |
| 0.7 | 0.2 | 0.1 | 0.1521 | 0.1604 | 0.4993 | 0.5193 | 0.0250 | 0.0260 |
| 0.8 | 0.1 | 0.1 | 0.1460 | 0.1550 | 0.4792 | 0.4992 | 0.0240 | 0.0250 |

Table A.10: Results. Combination of popularity with context (month) and cosine similarity (all) and matrix factorization.